

**A GRID META-SCHEDULER USING COMMUNITY SCHEDULER
FRAMEWORK (CSF) WITH DIFFERENT PLUG-IN POLICIES
PERFORMANCE EVALUATION**

BY


TAN ZHEN LING

**Thesis submitted in partial fulfillment of the requirements
for the degree of
Master of Science**

JUNE 2008

DECLARATION

This dissertation is the result of my own work except where specifically indicated in the text. I am aware that the degree awarded will be forfeited in the event of plagiarism.

Signature:

Date: 2nd July 2008

Name: (Tan Zhen Ling)

ACKNOWLEDGEMENTS

I would like to take this opportunity to convey my sincere gratitude to my supervisor, Dr. Chan Huah Yong. He has given this thesis his utmost attention and provided many constructive and invaluable suggestions to my research work.

I'd like to convey my heartfelt appreciation to School of Computer Science for providing a conducive and environment during the course of my research.

To my course mates in Grid Computing Lab, especially Imran, Ahmed, Adel and Ali, I'd like to express my thanks for them for their advice and friendship. Last but not least, I'd like to thank my family and friends for their encouragement and support.

TABLE OF CONTENTS

	Page
DECLARATION	ii
ACKNOWLEDGEMENTS	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	vi
LIST OF FIGURES	vii
ABSTRAK	ix
ABSTRACT	xi
CHAPTER 1 – INTRODUCTION	
1.1 Overview of Grid.....	1
1.2 Scheduler in Grid.....	2
1.3 Problem Statement.....	3
1.4 Scope and Objective	4
1.5 Contribution.....	5
1.6 Dissertation Outline.....	6
CHAPTER 2 – LITERATURE REVIEW	
2.1 Overview.....	8
2.2 Grid Computing.....	8
2.2 Grid Resource Management System (GRMS)	10
2.2.1 Grid Meta-Scheduler vs. Grid Local Scheduler.....	10
2.2.2 Grid meta-scheduler.....	12
2.2.2(a) Deployment Considerations.....	15
2.2.2(b) Scheduler Dependencies	16
2.2.3 Local Resource Manager	17
2.3 Community Scheduler Framework (CSF).....	20
2.3.1 Queuing Service.....	22
2.3.2 Job Service.....	23
2.3.3 Reservation Service	24
2.3.4 RM Adapter.....	25
2.4 Globus Toolkit	25
2.4.1 GridFTP	26
2.4.2 RSL	27
2.4.3 GRAM	29
2.4.4 Grid Information Service (GIS).....	30
2.5 Plug-in	31
2.6 Grid Scheduling Policy.....	32
2.6.1 Resource Selection Policy in Grid Scheduling.....	32
2.6.1(a) Resource Selection Based on Randomization	33
2.6.1(b) Resource Selection Based on Round-robin Manner.....	33
2.6.1(c) Resource Selection Based on CPU Utilization.....	34

2.7	Discussion.....	35
-----	-----------------	----

CHAPTER 3 – METHODOLOGY

3.1	Overview.....	37
3.2	Flow of Grid Scheduling	38
3.3	CSF Scheduling Plug-in	40
3.5	Job Queue and Resource Selection Policy.....	42
3.5.1	First-come-first-serve (FCFS) Queue	42
3.5.2	Resource Selection based on Round-robin manner	44
3.5.3	Resource Selection based on the Highest CPU Idle Percentage.....	45
3.5.4	Random Selection	47
3.6	Evaluation Method.....	49
3.6.1	CPU Utilization.....	49
3.6.2	Completion Time	50

CHAPTER 4 – IMPLEMENTATION

4.1	Overview.....	51
4.2	CSF Scheduling Plug-in API	51
4.3	Interface Method of CSF Scheduling Plug-in	52

CHAPTER 5 – TESTING, RESULTS AND DISCUSSION

5.1	Overview.....	53
5.2	Test-Bed Setup.....	53
5.3	Testing Scenario.....	56
5.3.1	CPU Availability	56
5.3.2	Scheduling Time	61
5.4	Qualitative Discussion	63
5.5	Evaluation Discussion	64

CHAPTER 6 – CONCLUSION AND FUTURE WORK

6.1	Overview.....	65
6.2	Summary of Work Done	65
6.3	Summary of Contribution.....	66
6.4	Objective Achieved.....	67
6.5	Future Work	68

LIST OF TABLES

		Page
Table 2.1	Current Implemented Grid Meta-scheduler	13
Table 2.2	Licensing and support	15
Table 2.3	Supported local resource manager	16
Table 2.4	Existent scheduling plug-in in CSF	23
Table 2.5	Advantages and disadvantages of different resource selection policies	35
Table 4.1	CSF interface class for scheduler plug-in	52
Table 5.1	Additional software required on CSF server and each node	55

LIST OF FIGURES

		Page
Figure 2.1	Grid scheduling architecture	11
Figure 2.2	Current implemented CSF architecture (Picture from (Community Scheduler Framework (2007)))	21
Figure 2.3	RSL job request specifying executable, arguments, input and error files	28
Figure 2.4	RSL job request specifying resources needed	28
Figure 3.1	Overall picture of job scheduling on Grid	38
Figure 3.2	Architecture of CSF job scheduling	40
Figure 3.3	General Flow of CSF scheduling plug-in modules	41
Figure 3.4	First-come first-serve (FCFS) job queue	43
Figure 3.5	Round-robin scheduling and tasks allocation	44
Figure 3.6	Pseudocode of resource selection based on round-robin manner	45
Figure 3.7	The highest CPU idle percentage scheduling and tasks allocation	46
Figure 3.8	Pseudocode of resource selection based on the highest CPU idle percentage	47
Figure 3.9	Randomization scheduling and tasks allocation	48
Figure 3.10	Pseudocode of random selection	48
Figure 4.1	Standard CSF plug-in API	51
Figure 5.1	CSF test-bed	54
Figure 5.2	CSF test setup	54
Figure 5.3	CPU information of cluster grid001	56

Figure 5.4	CPU information of cluster grid007	57
Figure 5.5	CPU utilization with different jobs scheduling policies	57
Figure 5.6	Round-robin job scheduling	58
Figure 5.7	Best CPU selection job scheduling	59
Figure 5.8	Number of jobs completed on each of the cluster by using Round-robin, the best CPU selection and Random selection policies	60
Figure 5.9	Comparison for jobs submission time with different scheduling plug-in policies	62

PENILAIAN PRESTASI PENJADUALAN META GRID YANG MENGGUNA COMMUNITY SCHEDULER FRAMEWORK (CSF) DENGAN PELBAGAI POLISI PLUG-IN

ABSTRAK

Community Scheduler Framework (CSF) adalah salah satu penjadualan meta grid semasa yang membolehkan penggunaan Globus. Penjadualan meta menjadualkan tugas bagi pihak penggunanya. Tujuan utama CSF adalah untuk menyediakan API yang mudah bagi penggunanya menentukan polisi masing-masing yang digunakan dalam servis giliran. Ia berkemampuan dengan menawarkan mekanisme penjadualan yang mudah dan membolehkan penambahan penjadulan plug-ins.

Adalah didapati bahawa CSF semasa mengandungi hanya satu penjadualan polisi plug-in yang berfungsi dengan cara round-robin semasa melakukan penjadualan tugas. Tesis ini menfokuskan dalam penilaian prestasi penjadualan polisi-polisi plug-in yang dapat bersatu dengan CSF. Oleh sedemikian, tesis ini mencadangkan dua jenis penjadualan polisi (pemilihan sumber yang berdasarkan purata peratus CPU tidak berguna yang tertinggi dan pemilihan sumber secara rawak) yang dapat bersatu dengan CSF untuk menjalankan penjadualan tugas.

Keputusan yang didapati daripada eksperimen membuktikan bahawa penambahan polisi penjadualan membolehkan CSF untuk menjalankan penjadualan genting dengan pemilihan sumber yang berdasarkan purata peratus CPU tidak berguna yang tertinggi, iaitu dimana peratus ketidakgunaan CPU adalah 70% lebih

besar yang terdapat daripada kutipan sumber. Masa keseluruhan yang digunakan untuk melengkapkan penjadualan untuk tugas berjumlah 40 yang bersaiz sama dengan menggunakan penjadualan cara round-robin adalah sebanyak 750 saat. Sebanyak 490 saat diperlukan dengan menggunakan pemilihan sumber yang berdasarkan purata peratus CPU tidak berguna yang tertinggi. Pemilihan sumber secara rawak pula memerlukan masa sebanyak 390 saat untuk melengkapkan penjadualan tugas-tugas tersebut. Masa yang diambil dengan menggunakan pemilihan sumber yang berdasarkan purata peratus CPU tidak berguna yang tertinggi adalah tinggi berbanding dengan pemilihan sumber secara rawak kerana aplikasi tambahan telah dipasang untuk mendapatkan informasi yang terperinci daripada kelompok yang tertentu supaya tugas yang seterusnya boleh dijadualkan. Untuk pemilihan sumber secara rawak, penjadual meta membahagikan sumber-sumber kepada tugas-tugas tanpa keperluan aplikasi tambahan.

A GRID META-SCHEDULER USING COMMUNITY SCHEDULER FRAMEWORK (CSF) WITH DIFFERENT PLUG-IN POLICIES PERFORMANCE EVALUATION

ABSTRACT

Community Scheduler Framework (CSF) is one of the current implemented grid meta-scheduler, which is Globus enable. The meta-scheduler schedules jobs on behalf of its users. The main objective of CSF is to provide simple API for writing user-defined policies for the queuing service. It has the ability to offer simple scheduling mechanisms and is extensible via scheduling plug-ins.

It is found that the current implemented CSF consists of only one scheduling plug-in policy where the submitted jobs are scheduled in round-robin manner. This thesis focuses on the performance evaluation of different scheduling plug-in policies that could be incorporated with CSF. This thesis proposed two scheduling plug-in policies (the highest average percentage of idle CPU selection and randomization resource selection) that incorporate with CSF for job scheduling.

The experimental results prove that by adding the highest average percentage of idle CPU selection plug-in, CSF is able to do the critical scheduling to the best resource with the highest percentage of CPU idles, which the percentage of CPU idles is greater than 70% in the resource pool. The completion time for the allocation of 40 jobs by using round-robin scheduling is about 750 seconds. Where as, for the best CPU selection is 490 seconds. It took about 390 seconds for random resource selection to complete the execution of 40 jobs. The time taken in CPU selection is

higher than random resource selection because a small application is installed on each of the master host in order to get the detail information of the particular cluster in which the next job will be submitted to. For the random resource selection, the meta-scheduler randomly assigned resources to jobs where no extra step is needed.

CHAPTER 1 – INTRODUCTON

1.1 Overview of Grid

In distributed systems, machines have the ability to communicate with each other where data and resources are shared among all. Grid can be defined as a geographically distributed computation platform. Grid computing made up of a set of heterogeneous machines that contains different types of hardware, where users can access via a single interface. Dynamic grid allows scalability and can be expand at anytime. Grid is autonomous as it can potentially access the resources of different organizations located in different geographical locations.

Grid computing is an advancement of distributed computing. Unlike distributed computing, grid computing focuses on large-scale resource sharing, offers applications innovation, and sometimes focuses on improvement of system performance to reach optimization. Grid enforces different computers to work together as one. With this, it is possible that one's can reach out and use the computational or storage resources on machines other than their own. By doing so, users can share their own resources with others, and hence, the utilization of unused resources will be increased.

Grid computing is an overlapping of both distributed and parallel computing. Grid computing is said to be distributed because of its heterogeneous nature. While

distributing, grid can work in parallel where few resources working together on one task. The key concept of grid computing (Talia (2002)) is to provide resource access and operational services across widely distributed virtual organizations.

Virtual organizations are highly dependent on grid. They may share some common requirements but may vary in size and structure, which made up of the combination of servers, mainframe computers, client hosts and so on. The members of any virtual organization negotiate on resource sharing based on the rules and conditions defined, in order to share the resources from the automatically constructed resource pool (Joseph and Craig (2004)).

1.2 Scheduler in Grid

In the early grid, the administrators need to manually configure and submit the jobs. The administrators required to have the knowledge and direct control of all the jobs on the grid as they will execute jobs manually based on their scheduling policy. When the size of the grid environment increasing, the number of job increase as well. Due to this, more resources is needed. Effective scheduling is important in optimizing resource usage.

Nowadays, jobs scheduling in a cluster is done by using the scheduler or local scheduler. A scheduler is responsible to carry out the scheduling tasks. A user does not need to manually coordinate the access to the resources as the scheduler has the

function of automatically select and schedule resource allocation for jobs.

Tasks scheduling (Subramani et. al. (2002)) become more complex while it is in a meta-computing environment since many clusters with different local scheduling policies are involved. Grid schedulers as in (Bart et. al. (May 2003)) use measurement information about the current utilization of machines to determine which ones are not busy before submitting a job and are usually react to the immediate grid load.

Grid meta-scheduler is always dynamic, heterogeneous and diverse, and has to deal with local issues. It is responsible on monitors the progress of scheduled jobs managing the overall work-flow. The scheduler will automatically re-allocate the jobs if the jobs are lost due to system or network outages. However, if a job appears to be in an infinite loop and reaches a maximum timeout, then such jobs should be dropped and should not be re-scheduled.

1.3 Problem Statement

Community Scheduler Framework (CSF) is one of the current implemented grid meta-scheduler, which is Globus enable. CSF can submit a job, make an advance reservation for resources, query the status of a particular job, retrieve information regarding the current schedule, and remove a job which the meta-scheduler has submitted. The meta-scheduler schedules jobs on behalf of its users.

But, it is found that the current implemented CSF consists of only one scheduling plug-in policy that incorporate with CSF where the submitted jobs are scheduled in round-robin manner. Due to the limited number of scheduling plug-in policy provided by CSF, it needs more scheduling policies in order to improve the robustness of CSF as a grid meta-scheduler.

1.4 Scope and Objective

The scope of this dissertation is to analyze and evaluate the performance of grid meta-scheduler using Community Scheduler Framework (CSF) with different scheduling plug-in policies.

The main objective is to enable CSF to perform critical jobs scheduling. In order to increase the utilization of idle resources in the resource pool, a scheduling plug-in policy based on the best resource selection will be proposed. On the other hand, a random selection policy will appear to be the second scheduling plug-in policy in resource selection for jobs allocation.

1.5 Contribution

After in-depth review of CSF related work, it shows that CSF is only provides round-robin scheduling policy in allocating jobs to resources.

In this dissertation, we propose two different scheduling plug-in policies that incorporate with CSF by using the default round-robin scheduling plug-in policy as the benchmark. The first proposed scheduling plug-in policy incorporate with CSF is a resource selection scheduling plug-in policy base on the best CPU selection. The second proposed CSF scheduling plug-in policy is randomized resource selection.

Secondly, we evaluate and analyze the performance of CSF with the scheduling plug-in policy of round-robin manner, the best CPU selection policy and randomization selection policy base on the resource utilization and the completion time. Additionally, the scalability of CSF in job scheduling and the easiness of adding new plug-in will be taken into account.

1.6 Dissertation Outline

The outline of this dissertation is as follow:-

Chapter 1

This chapter presents the overall introduction of the study. This chapter covers also the problem statement, scope and objective of, and the research contribution.

Chapter 2

Literature review section. In this chapter, researches need to be done on topics that are related to grid meta-scheduler. Review some of the field that related or relevant to this topic. We will also review on what a meta-scheduler is and its functionality. Some of the current implementation will be discussed in this section as well so that we are able to know the current trend of the meta-scheduler.

Chapter 3

Design methodology of the CSF scheduling plug-in will be discussed in this chapter. This chapter will also presents the idea of how the grid meta-scheduler be implemented in grid environment.

Chapter 4

This particular chapter will discuss on the implementation of CSF scheduling plug-in policies.

Chapter 5

This chapter discuss on the results and the outcomes of job scheduling in comparison of different scheduling plug-in policies to the grid meta-scheduler. What is important in order to get the expected results and justification of the results will be discussed.

Chapter 6

Finally, chapter 6 conclude this research work by providing the summary and the future work.

CHAPTER 2 – LITERATURE REVIEW

2.1 Overview

In this section, we conducted the literature review with respect to our field of study. The section proceeds with the brief overview and concepts of grid computing along with the key explanation of currently implemented Grid meta-scheduler.

2.2 Grid Computing

Grid computing is an important field in the computer industry and is becoming more popular as time goes on. The dramatically increase of the complexity of the computational tasks require higher performance environment to solve the computational problems.

Looking at the problem above, resources required to accomplish such tasks have increased, in order to get a higher performance problem solving environment. Resources required refer to the computing power, data storage, computational machines, such as mainframe computers, server and client hosts, and etc. Problem can be solved by increasing the number of resources that are needed. But, it will be very costly for an organization to support large number of resources.

Computational grid (Debral (2006)), which is a large-scale dynamical network of geographically distributed peer resources clusters, can be one of the solutions to solve this problem. It made up of different computing hosts, such as high performance computing clusters, desktop and etc. Grid computing (Lu Bingfeng et. al. (2005)) will operate by integrating the available resources that to form a virtual organization comprised to multiple resources associated with heterogeneous clusters.

Below are four main characteristics of a Grid (Buyya (2002)):

a) Multiple Administrative Domains and Autonomy

Grid resources are geographically distributed across multiple administrative domains and owned by different organizations.

b) Heterogeneity

A Grid involves a multiplicity of resources that are heterogeneous in nature and will encompass a vast range of technologies.

c) Scalability

A Grid might grow from a few integrated resources to millions and may lead to the degradation of the systems performance. Thus, applications designed must be latency and bandwidth tolerant.

d) Dynamicity or Adaptability

In a Grid, the probability of resource failing is high. Resource managers or applications must tailor their behavior dynamically and use the available resources and services efficiently and effectively.

2.2 Grid Resource Management System (GRMS)

The Grid Resource Management System (GRMS) (Andrea et. al. (2005)) responsible to find and allocate feasible resources, for example, CPU cycles, storage, bandwidth, and etc in order to satisfy the user request. The GRMS then monitors the correct task processing, and notifies the user when the results are available. The GRMS must be able to utilize the residual resources of each organization being part of the grid.

2.2.1 Grid Meta-Scheduler vs. Grid Local Scheduler

Resource balancing in grid environment is important. The goal of resource balancing is to improve the performance of the system by optimizing the throughput or minimizing the costs. Therefore, grid middleware is introduced to schedule resources across the heterogeneous distributed infrastructure. A grid scheduler has the ability to exploit idle resources to maximize the resources availability.

A Grid computing environment provides the virtual computing resource that can be used to execute applications. In a Grid system (Mausolf (2005)), the resource managers coordinate and control local resources while the meta-scheduler operate at the grid level supervise the resource managers. It manages jobs by doing resource allocation and resource re-allocation if necessary by gathering and analyzing information from local resource managers in order to assign user jobs to the most

suitable resources at any given time.

The grid meta-scheduler (Andrea et. al. (2005)) is in charge of dividing the job into a number of tasks and allocating each task to a cluster. At each cluster, a local scheduler is responsible for determining job sequencing, local resource allocation and data transfer scheduling.

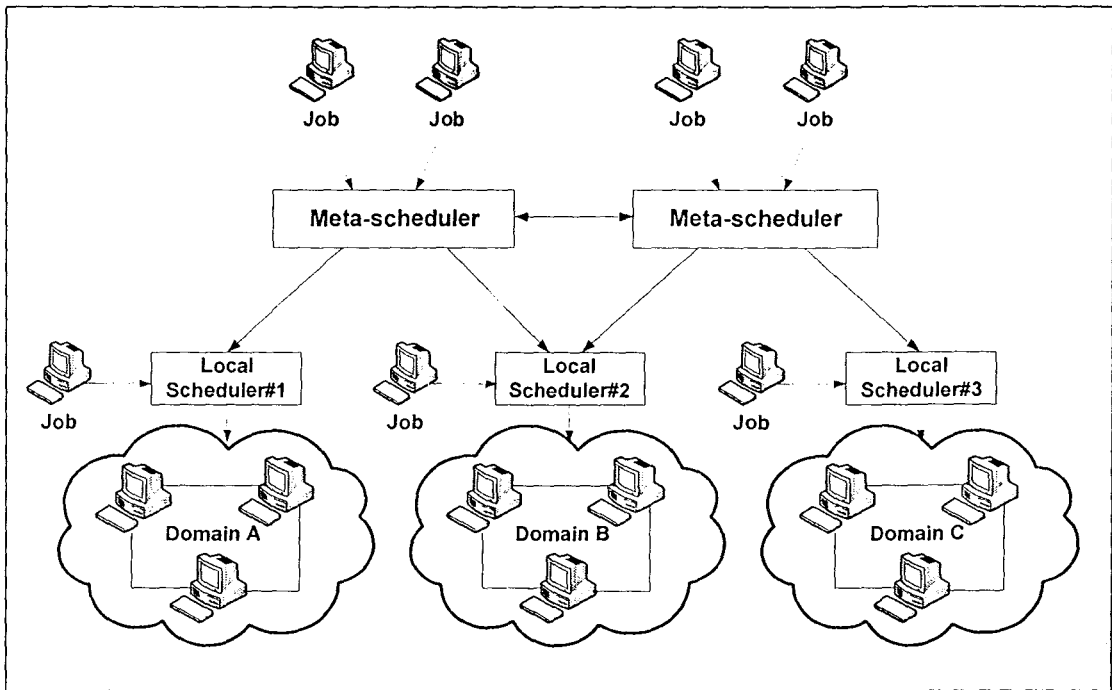


Figure 2.1: Grid scheduling architecture

When a suitable site is located, the task request is passed from the meta-scheduler to the selected local scheduler. From Figure 2.1, a grid meta-scheduler is operating on top of local schedulers. Job is submitted to the meta-scheduler. The job will then be submitted to the lower level schedulers or a cluster

scheduler by the grid meta-scheduler based on define scheduling policy, which used by the grid meta-scheduler.

2.2.2 Grid meta-scheduler

A meta-scheduler (Yan and Chapman (2006)) consists of functions of *scheduling* and *Meta*. For scheduling, it has the ability that allows co-allocating resources for application requiring collaboration between multiple sites. The Meta has the capability to negotiate with local schedulers to satisfy global Grid requests.

A grid meta-scheduler is the resource management component coordinating access to the Grid resources. Meta-scheduling (Adzigogov et. al. (2005)) is a technology in the Grid that is responsible for managing jobs and application work flow, including submitting, scheduling, executing, monitoring, stopping, and retrieving results of computational jobs. Scheduling is one of the ways for maximizing the resources availability to end users. At the same time, it can be used to exploit idle resources. It is responsible for the balancing the work load between sites and data centers.

The scheduler is responsible for data management and providing services for tasks in management level. For instances (Joseph and Craig (2004)), advance resource reservation, job and resource policy management and enforcement for best turnaround times within the allowable budget constraint, monitoring job executions

and status and so on.

A computational grid with a grid meta-scheduler allows the user to authenticate with a single system to be able to submit jobs in a uniform manner. The system determines where the jobs should run to complete in the shortest amount of time while adhering to business policies and maintaining service-level agreements (SLAs). On the other hand, grid meta-scheduler also makes decisions based on job requirement or QoS requirement to assign jobs.

Table 2.1: Current Implemented Grid Meta-scheduler

Grid Meta-scheduler	Description
CSF & Platform CSF Plus	An open source framework for implementing a grid meta-scheduler, with the use of the Globus Toolkit middleware. CSF was developed by Platform Computing in cooperation with the Jilin University, China.
Grid Service Broker	Developed in the Grid Computing and Distributed Systems Laboratory at the University of Melbourne, Australia.
GridWay Metascheduler	A lightweight meta-scheduler developed by a team working for Distributed Architecture Group from Universidad Complutense in Madrid, Spain
Moab Grid Scheduler (Silver)	A feature-abundant meta-scheduler developed by Cluster Resources Inc.
EGEE Workload Manager Service (WMS)	Developed as a part of the Enabling Grids for E-science (EGEE) project funded by the European Commission
Nimrod/G & Axceleon EnFuzion	Developed at the Monach University, is a specialized modeling system that uses simple declarative language to express parametric experiments.
MP Synergy	A product of United Devices, is designed for virtualized management of the entire enterprise infrastructure.
Condor-G	A fault-tolerant job submission system that can access various computing resources, and scheduling has to be implemented above Condor-G.

The existing the meta-scheduler that can be found (Refer to Table 2.1), such as Community Scheduler Framework (CSF), Grid Service Broker, GridWay Metascheduler, Moab Grid Scheduler (Silver), EGEE Workload Manager Service (WMS), Nimrod/G, MP Synergy and Condor-G.

From Table 2.1, we would like to remark the following:

- CSF (Community Scheduler Framework (2007)) supports advance reservation booking and offers round-robin and reservation based scheduling algorithms.
- GridWay Metascheduler (GridWay Metascheduler (2008)) provides Globus user with a grid scheduling functionality similar to that found in local DRM (Distributed Resource Management) systems.
- Nimrod/G (Nimrod/G (2007)) strives for the equilibrium between resource providers and resources consumers via auctioning mechanisms.
- EGEE Workload Manager Service (WMS) (V'Azquez-Poletti et. al. (2006)) provides a higher centralized scheduling strategy at VO-level if compared to GridWay Metascheduler (site-level scheduling).
- Condor-G (Condor-G (2007)) does not support scheduling policies, but, it supplies mechanisms, such as ClassAd and DAGMan that may be useful for a meta-scheduler standing above.

2.2.2(a) Deployment Considerations

Deployment cost is an important consideration while selecting a scheduler.

Deployment costs are directly dependent on a number of factors.

Table 2.2: Licensing and support

Grid Meta-scheduler	Licensing	Support
CSF & Platform CSF Plus	Free	Free
Grid Service Broker	Free	Free
GridWay Metascheduler	Free	Free
Moab Grid Scheduler (Silver)	Commercial	Commercial
EGEE Workload Manager Service (WMS)	Free	Free
Nimrod/G & Axceleon EnFuzion	Free for Nimrod/G Commercial for EnFuzion	Free for Nimrod/G Commercial for EnFuzion
MP Synergy	Commercial	Commercial
Condor-G	Free	Free

The most direct cost is the cost of licensing and support from the scheduler developer. These factors are evaluated in Table 2.2. Most of the schedulers are free and open source, except for MP Synergy and Moab, which are commercial products. Since the meta-scheduling products are still evolving, it is likely that changes to the scheduler will be required, which might lead to more costs, especially if the code is not open-source.

2.2.2(b) Scheduler Dependencies

The ability to integrate the scheduler with existing local resource managers such as PBS, LSF, and SGE are given in Table 2.3.

Table 2.3: Supported local resource manager

Grid Meta-scheduler	Local Resource Managers Supported
CSF & Platform CSF Plus	GRAM(WS&Pre-WS), LSF, OpenPBS, Condor, SGE
Grid Service Broker	GRAM(WS&Pre-WS), Condor, PBS, SGE
GridWay Metascheduler	GRAM(WS&Pre-WS), SGE ('GE-GT adapter' is needed in order to interact between GridWay and SGE), LSF, PBS
Moab Grid Scheduler (Silver)	GRAM(WS&Pre-WS), Torque, PBS, LSF, LoadLeveler
EGEE Workload Manager Service (WMS)	GRAM(Pre-WS), Condor
Nimrod/G & Axceleon EnFuzion	Globus Toolkit, Legion, Condor, Grid Engine, LSF, PBS
MP Synergy	GRAM(Pre-WS), LSF, OpenPBS, PBS Pro, SGE, LoadLeveler, Condor
Condor-G	GRAM(WS&Pre-WS)

There are few factors that are taking into consideration in this evaluation. Firstly, scheduler supports for job submission/management to the local resource managers currently deployed on Grid Computing Lab resources. Currently supported local resource manager in Grid Computing Lab is SGE. Then, second factor is whether the scheduler restricts the usage of the resource by (possibly local) users by submitting jobs directly to the local resource managers.

From Table 2.2 and Table 2.3, we can conclude that, CSF, Grid Service Broker and GridWay Metascheduler are more suitable than others to deploy in Grid Computing Lab due to the reason of the supported local resource manager and licensing free. MP Synergy is not taking into the consideration as it is a commercial product.

2.2.3 Local Resource Manager

Distributed Management Systems (DRMs), also known as workload management systems, provide resource management for jobs that are submitted to run on any given resource. There is some current implemented resource manager that can be integrated with the CSF working on the Globus environment.

a) Portable Batch System (PBS)

PBS (Bayucan et. al. (1996)), (Henderson and Tweten (1995)) is a package which was designed and written by the Numerical Aerodynamic Simulation Complex, NASA, from 1994. PBS is a successor to NQS, and is able to address many of the deficiencies of NQS. It was designed to provide additional controls over the initiating, or scheduling, of execution of batch jobs.

PBS extends the UNIX operating system by the addition of user-level services. PBS had detailed design and implementation documentation for developers. It was designed to be easy to add functionality and improved over NQS in a number of ways, including the provision for parallel jobs. PBS also included features that allow the scheduling policy to be modified according to a site's needs. A batch scheduling language has been designed for use with PBS, but the documentation recommends that it not be used, as there are implementation problems.

b) Load Sharing Facility (LSF)

LSF (Platform Computing (2008)) by Platform Computing, is a very popular commercial batch queuing system. LSF relies on the existence of shared files to implement queues, locks and logs. On the other hand, LSF has some measure of fault tolerance inbuilt. If a shared file system is not available, the degree of fault tolerance is reduced. If the master host that makes scheduling decisions fails, another host in the cluster is automatically voted to be the master. Hosts are elected to be the master in the order they appear in a static file which must be visible to all machines in the cluster. While the cluster becomes partitioned by network failure, the partition that has access to the LSF log files continues working, while the remaining partitions sit idle.

c) Condor

Condor (Joseph and Craig (2004)) is well suited for parameter studies and high throughput computing, where jobs generally do not need to communicate with each other. It can be classified as a specialized workload management system for computation-intensive jobs. Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Upon receiving serial or parallel jobs from the user, the Condor system places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.

d) Sun Grid Engine (SGE)

SGE (Sun Microsystems (2008)) is the foundation of the Sun Grid utility computing system. It is typically used on a computer farm or computer cluster and is responsible for accepting, scheduling, dispatching, and managing the remote execution of large numbers of standalone, parallel or interactive user jobs. It also manages and schedules the allocation of distributed resources such as processors, memory, disk space, and software licenses.

2.3 Community Scheduler Framework (CSF)

The Community Scheduler Framework (CSF) (Community Scheduler Framework (2008)) is a set of Grid Services developed by Platform Computing in cooperation with Jilin University, China, which is implemented using the Globus Toolkit. Globus is a software for grid. It is an open source tool that is used to build applications that exploit grid infrastructure. CSF is a set of modules that can be assembled to create a meta-scheduling system that accepts job requests and executes them using available grid compute services.

The main objective of CSF is to provide simple API for writing user-defined policies for the queuing service. Besides, the implementation of CSF allows reservations on resources. CSF has the ability to offer simple scheduling mechanisms and is extensible via scheduling plug-ins.

CSF supports the emerging WS-Agreement specification and the Globus Toolkit's GRAM service. It also uses the WS Core monitoring mechanisms and the Globus Index Service. CSF provides an extensible framework (Shread (2003)) for implementing meta-schedulers that can negotiate with heterogeneous workload execution software such as Platform LSF to acquire the right resources to fulfil computing requirements.

Basic functionalities provided by CSF are submitting jobs to Grid without specifying Cluster, monitoring and control jobs, provide Queuing Service, schedule

jobs and resource by custom-built policies and a CSF Portlet which is a web browser based User Interface.

Advanced functionalities provided by the current version of CSF are Multiple Domains Resources Information Sharing, automatic user credentials delegation, automatic data-staging, extensible scheduling policies, and supporting grid parallel jobs (MPICH-G2).

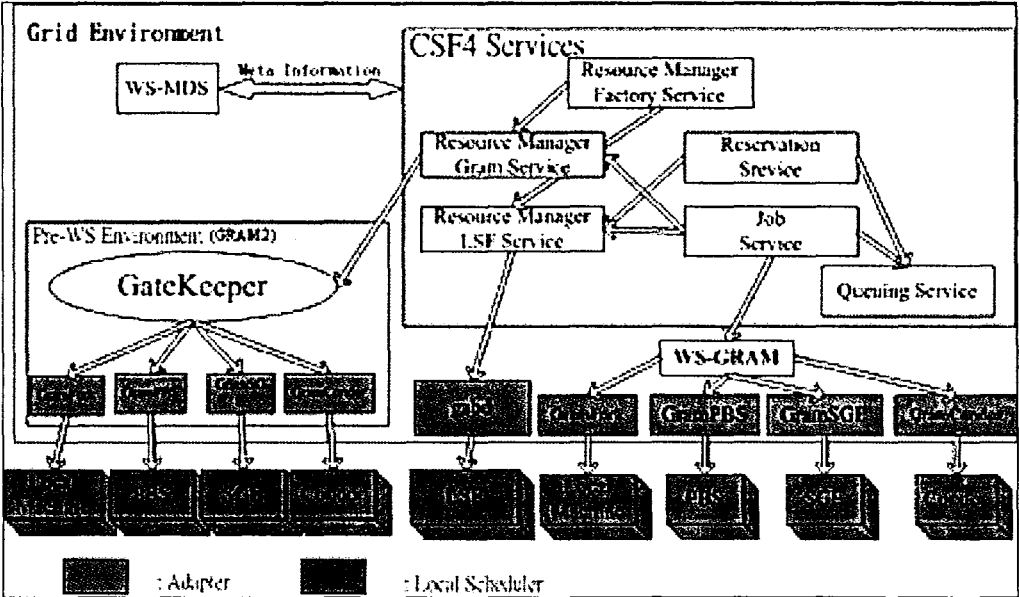


Figure 2.2: Current implemented CSF architecture
(Picture from (Community Scheduler Framework (2007)))

Figure 2.2 shows the architecture of the current implemented CSF version. CSF enables communication between heterogeneous schedulers on local level. Different local schedulers at different clusters, which contains of different types of machines

running on it can now communicate with each others by using CSF. The list of supported schedulers includes Platform Load Sharing Facility (LSF), Open Portable Batch System (Open PBS), Grid Engine (SGE / N1GE) and Condor.

CSF contains of the components below (Platform Computing (2006)):

- **Job service** – Creates, monitors and controls compute jobs
- **Reservation service** – Guarantees resources are available for running a job
- **Global Information Service** – Allows for the propagation of information between resource managers and the meta-scheduler
- **Queuing service** – Provides a service where administrators can customize and define scheduling policies at the VO level, and/or at the different resource manager levels
- **Resource Manager Adapter Service (RM Adapter)** – Provides Grid service interface which bridges the Grid service protocol and resource managers

2.3.1 Queuing Service

The Queuing Service provides scheduling capabilities, where the grid meta-scheduler maps jobs to resource managers based on defined policies at the VO level. If there are multiple scheduling policies in the Queuing Service, the different schedulers will be called in order based on the job list and on the job decisions,

allowing the effects of one scheduling plug-in to be combined with the effects of another.

Table 2.4: Existent scheduling plug-in in CSF

Scheduling Plug-in	Description
FCFS Round-Robin	Default scheduling plug-in
Throttle	Restricts the maximal number of jobs can be dispatched in a scheduling cycle
Array Job Plug-in	Design for life science applications (such as AutoDock, BLAST)
MPICH-G2 Plug-in	By using VJM, the plug-in guarantee the synchronized resource allocation can be successful

The CSF Queuing service provides FCFS queue and round-robin, job throttle, array job plug-in and MPICH-G2 plug-in (Refer to Table 2.4). In such a case, there is only one scheduling plug-in policy that is incorporated with CSF currently, which is the round-robin scheduling.

2.3.2 Job Service

A Job Service provides an interface for placing jobs on a resource manager and interacting with the job once it has been dispatched to the resource manager. The Job Service provides basic matchmaking capabilities between the requirements of the job and the underlying resource manager for running the job. The Job Service uses information such as policies, which are defined at the meta-scheduler level, and

resource information about available resource managers, queues, host, and job statuses as provided by the Global Information Service.

In the current version of the CSF, the Job Service accepts client requests in the form of the Globus Resource Specification Language (RSL). A given Job Service instance can actually manage multiple user jobs. A Job Service Dispatcher component responsible for matching user job submission requests to run Job Service instances. The Job Service Dispatcher creates a Job Service instance on behalf of the user if there is no applicable Job Service available for the user.

2.3.3 Reservation Service

The Reservation service allows end-users or a Job Service to reserve resources under the control of a resource manager to guarantee their availability to run a job. This service allows reservations for any type of resource. Once a reservation is made, a Job Service sends a job to a resource manager that is associated to a provided reservation. It also allows request of a new reservation with a particular resource requirement, starting at a specific time for a given duration, remove a reservation and retrieve the details of a particular reservation.

The Reservation Service makes use of information about the existing resource managers and policies that are defined at the meta-scheduler level and will make use of a logging service to log reservations.